Mapper pseudo code for emitting word count

```
Class mapper
        For all lines in standard input:
                Year <- current file name input
                Line <- line cleaned of punctuation and html tags

                Words <- new list created from the line

                For each word in words
                        Emit(word, 1, year)
```

Reducer class for counting the average use of word per year along with maximum and minimum

```
Class reducer
        Word2count <- new Dictionary (the key will be a tuple of (word,year)

        For all lines in standard input:
                Word, count, year = new list created off input line
                Try:
                        Count = count as integer

                Try:
                        Add to count value for year_count dictionary
                Except:
                        Initialize year_count dictionary for year if it does not exist

                Try:
                        Add count to word count dictionary, with key  of current word, year
                Except:
                        Initialize word count value for word, year to the current count

        Initialize two dictionaries for word_max and word_min

        For each (word, year) key  in word2count dictionary:

                Try:
                        If the current max for  word is less than the count for word (obtained
from word2count[(word,year)]:
                                Set current word max value to the count for (word,year)
                Except:
```

The word does not have a max set. Initialize word max dictionary to the wordcount for the word and year

Try:
If the current minimum word is greater than the count for the (word,year):
Word min count for this word is set to the count for (word,year)
Except:
The word does not have a min set. Initialize word min dict for word to wordcount for word, year

For each (word,year) key in the word2count dictionary:
Emit(word, wiord count value, year, average that year (word count for that year divided by total number of words for that year, taken from year_count dict), wordmax for word, and wordmin for word

Reducer class for calculating window avg, stdev, and changes between windows
Class reducer:
method getWindowWordCount(word,window_start_year):
Return number of times a word appeared in the window (window starts at window_start_year)

Method getStd(word, start_year):
Try:
Word_count_list = number of times a word appeared in each year of the window
Except:
Word did not appear in dictionary, append 0 to word_count_list

Return standard deviation (word_count_list)

Word2count <- new dictionary
Year_count <- new dictionary
Totalwordcount = 0
Rangecount = new dictionary

For each line in standard input
Word,year,line = line.split on whitespace
Total count = total count + 1

Try:
Count = count converted to an integer

```
            Try:
                Add count to the count value for year_count[for this year]
            Except:
                Initialize year_count for year to count value

            Try:
                Add count to the word2count dict for tuple key (word,year)
            Except:
                Initialize word 2 count dict for tuple key (word,year) to count

        Window_Year = 1984
            While year is less than 2017:
            For each tuple key (word, year) in word2count keys:
                If the year key is greater than  window_year and less than or equal to
window_year + 4:
                Window_word_cont = getWindowWordCount(key word, window_year)

                Avg = window_word_count / 4.0 (number of years in a window)

                Std = getStd(word key, window_year)

                Emit(avg, std)

                If key year is equal to end of window:
                    Next_year = window_year + 1

                    Try:
                        Word_count_for_next year = word2count[key word,
nextyear]
                        Threshold = avg + ( std * 2.0)

                        If word_count_for_next_year > threshold:
                            Emit ('threshold exceeded ' word, next_year_count
next year)


            If year is not equal to 2016:
                Add 4 to the year
            Else year equals 2016:
                Add 1 to year (no speeches beyond 2018)
```